Attorney Docket No.: MP0393

UNITED STATES PATENT APPLICATION

FOR

MEMORY MAPPED REGISTER FILE AND METHOD FOR ACCESSING THE SAME

BY

Hong-Yi CHEN

AND

Henry Hin Kwong FAN

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

MEMORY MAPPED REGISTER FILE AND METHOD FOR ACCESSING THE SAME

FIELD

[001] This invention relates generally to data processing systems and, more particularly, to a memory mapped register file and method for accessing the same.

BACKGROUND

[002] General purpose registers or a "register file" are a useful component of a data processing system's processing architecture. For instance, a microprocessor or central processing unit (CPU) of a data processing system retrieves operands from one or more general purpose registers to execute instructions. The results of the executed instructions are then stored back in one of the general purpose registers. This allows the data processing system to execute instructions more efficiently. Many prior microprocessor architectures operate in different processor modes, and generally use general purpose registers designated, e.g., R0 through R15.

[003] Typically, prior microprocessor architectures, which use such general purpose registers, process instructions in a normal user mode and an exception handling mode, such as an interrupt mode. For example, in the interrupt mode, a user application is halted in response to an interrupt to a processor. Furthermore, in the interrupt mode, access to some general purpose registers is performed through a separate memory unit referred to as "banked registers" in order to improve exception handling processing. That is, in the interrupt mode, different registers in a separate memory unit, i.e., banked registers, are accessed than in the normal mode.

[004] **FIG. 1** illustrates "banked registers" for a prior art microprocessor architecture. As shown, in normal mode, a microprocessor (not shown) accesses registers R_0

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

through R_{15} in normal mode general purpose registers 100, whereas, in interrupt mode, the microprocessor accesses only registers R_8 through R_{15} in interrupt mode banked registers 102. In this manner, accessing registers in normal mode and interrupt mode requires access to separate memory units. A disadvantage, however, of using banked registers is that it requires a special type of naming scheme to distinguish between the general purpose registers and banked registers, which increases processing overhead. Furthermore, such a general purpose register or "register file" scheme inefficiently accesses registers by requiring access to separate memory units for different processor modes.

[005] Another disadvantage of using the prior art architecture of banked registers is that it does not provide flexibility and efficiency in accessing registers. In particular, the prior art architecture does not allow for aliasing, which is the ability to arbitrarily use the same register across different processor modes. Additionally, the prior art architecture does not allow for fragmentation in which registers can be located in discontinuous locations in memory for access in different processor modes.

[006] There exists, therefore, a need for improved general purpose registers or register files without using separate memory units for register access in different processor modes.

SUMMARY

[007] According to one aspect of the invention, a register file for a data processing system comprises a memory unit, input ports, and output ports. The memory unit includes a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register and processor mode. The input ports receive inputs for addressing at least one of the memory locations using an

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

encoded address. The output ports output data from at least one of the memory locations addressable by an encoded address.

[008] According to another aspect of the invention, a register file for a data processing system comprises memory means, input means, and output means. The memory means includes a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register means and processor mode. The input means receive inputs for addressing at least one of the memory locations using an encoded address. The output means output data from at least one of the memory locations addressable by an encoded address.

[009] According to another aspect of the invention, a data processing system comprises a processor to process instructions and a plurality of pipeline stages to execute instructions. The pipeline stages include a register file. The register file includes a memory unit, input ports, and output ports. The memory unit includes a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register and processing mode. The input ports receive inputs for addressing at least one of the memory locations using an encoded address. The output ports output data from at least one of the memory locations using an encoded address.

[010] According to another aspect of the invention, a data processing system comprises processing means for processing instructions that includes pipeline means for executing instructions. The pipeline means includes register file means. The register file means includes a memory means, input means, and output means. The memory means includes a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register means and

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

processing mode. The input means receive inputs for addressing at least one of the memory locations using an encoded address. The output means output data from at least one of the memory locations using an encoded address.

[011] According to another aspect of the invention, a processor comprises an integrated circuit that includes a memory unit and at least one address encoder. The memory unit includes a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register and processor mode. Each address encoder provides at least one encoded address for addressing at least one of the memory locations.

[012] According to another aspect of the invention, a data processing system comprises a memory mapped register file for accessing a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register and processor mode.

[013] According to another aspect of the invention, a processor comprises circuit means the includes memory means and addressing means. The memory means includes a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register means and processor mode. Each addressing means provides at least one encoded address for addressing at least one of the memory locations.

[014] According to another aspect of the invention, an integrated circuit method comprises configuring the integrated circuit to receive inputs; configuring the integrated circuit to determine an encoded address based on the received inputs, wherein the encoded address corresponds to at least one register and processor mode; configuring the integrated

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

circuit to access a register using an encoded address; and configuring the integrated circuit to output data from the accessed register.

[015] According to another aspect of the invention, a method for accessing a memory unit having a plurality of memory locations, the method comprises receiving a memory request for accessing the memory unit, the memory request including a register index input and a processor mode input; encoding the register index input and processor mode input to obtain an encoded address; accessing at least one of the memory locations of the memory unit in accordance with the encoded address, wherein the encoded address corresponds to at least one register and processor mode; and writing data into or reading data from the accessed memory location.

[016] According to another aspect of the invention, a memory unit comprises a plurality of memory locations addressable by encoded addresses, wherein each encoded address corresponds to at least one register and processor mode.

DESCRIPTION OF THE DRAWINGS

[017] The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate exemplary implementations and embodiments of the invention and, together with the detailed description, serve to explain the principles of the invention. In the drawings,

[018] **FIG. 1** illustrates banked registers for a prior art microprocessor architecture;

[019] **FIG. 2** illustrates one example of a data processing system having a pipeline microprocessor architecture with a register file;

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[020] **FIG. 3** illustrates in block diagram form one example of inputs and outputs for the register file of **FIG. 2**;

[021] **FIG. 4** illustrates a detailed circuit diagram of one example of the register file of **FIG. 3**;

[022] **FIG. 5** illustrates a block diagram of one example of an address encoder with mapping control logic and a mapping table for obtaining an encoded address to access a register or storage location in the register file of **FIG. 4**;

[023] **FIG. 6** illustrates a block diagram of one example of indices corresponding to general purpose registers;

[024] **FIG. 7** illustrates a block diagram of one example of a mapping table for the address encoder of **FIG. 5**;

[025] **FIG. 8** illustrates a block diagram of one examples of sixteen indices corresponding to sixteen general purpose registers;

[026] **FIG. 9** illustrates a diagram of one example of a mapping table for the indices of **FIG. 8**;

[027] **FIG. 10A** illustrates a diagram of one example of aliasing registers in the register file of **FIG. 4**;

[028] **FIG. 10B** illustrates a diagram of one example of aliasing and fragmenting the registers in the register file of **FIG. 4**;

[029] **FIG. 10C** illustrates a diagram of another example of fragmenting the registers in the register file of **FIG. 4**;

[030] **FIG. 11** illustrates one example of a flow diagram for a method to output source data from the register file of **FIG. 4**; and

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[031] **FIG. 12** illustrates one example of a flow diagram for a method to write data into the register file of **FIG. 4**.

DETAILED DESCRIPTION

[032] Reference will now be made in detail to exemplary implementations and embodiments of the invention, examples of which are illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

[033] A memory mapped register file is disclosed that overcomes disadvantages of prior register files and provides a more flexible and efficient manner of accessing registers in a register file of a data processing system that can operate in multiple processor modes. According to one example, a register file for a data processing system includes a memory unit, input ports, and output ports. The memory unit includes a plurality of memory locations. Each memory location is addressable by an encoded address, wherein the encoded address corresponds to at least one register and processor mode. The input ports receive inputs for addressing at least one memory location using an encoded address. The output ports output data from at least memory location addressable by an encoded address. In this manner, a register is addressable and thus accessible by using the memory location addressable by the corresponding encoded address.

[034] Thus, memory locations (corresponding to respective registers) can be accessed in the memory unit of the register file for different processor modes without using separate memory units such as "banked registers." By avoiding banked registers use, a more efficient manner of accessing registers in the register file can be achieved. Processing overhead is also improved by making each register (i.e., memory location) of the memory

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

unit addressable by an encoded address without requiring a special naming scheme for accessing registers in separate banked registers for different processor modes.

[035] Additionally, addressing techniques disclosed herein allow for flexible and efficient use of memory in the register file for accessing memory registers across different processor modes. In one example, the registers in the register file can be aliased to achieve convenient addressing of registers in limited memory space. For instance, aliasing allows the same register to be used in the memory unit of the register file across different processor modes. Thus, aliasing allows for sharing of data in an aliased register without having to transfer data between processor modes. In addition, aliasing allows a limited number of registers to be used for a data processing system operating in different processor modes.

[036] In other examples, registers can be fragmented within the register file such that registers for one particular processor mode are isolated and not used in other processor modes. This is particularly useful when operating in a less frequent processing mode such as emulation mode of a processor Addressing techniques disclosed herein also allow for registers to be scalable to any bit width without requiring extensive redesign of the register file. In particular, regardless of the bit width of the register, registers in the register file are addressable by the same encoded addresses, thus allowing for increased flexibility of a register file.

[037] In the following description, reference to a "memory mapped register file" is a register file having a memory unit with a plurality of registers corresponding to memory locations addressable by an encoded addresses, wherein each encoded address corresponds to at least one register and processor mode. Thus, a "register" is also addressable by the encoded address to its corresponding memory location. Such a register file can be

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

implemented for various types of microprocessor architectures that require access to registers in different processor modes.

[038] **FIG. 2** illustrates one example of a data processing system 200 having a pipeline microprocessor architecture with a register file 206. The pipeline architecture processes instructions in multiple stages. Data processing system 200 includes a pipeline comprising an instruction fetch stage 202, an instruction decode stage 204, a register file 206, an execution stage 208, and a write back or retire logic stage 210. Data processing system 200 also includes a program memory 212 that stores instruction data used by execution stage 208. In this example, the pipeline stages and memory can be implemented as an integrated circuit (IC) using any combination of circuits, components, memory devices, and bus lines.

[039] As will be described in further detail below, register file 206 is a memory mapped register file having a memory unit with a plurality of registers corresponding to memory locations addressable by an encoded address. Each encoded address corresponds to at least one register and processor mode. Register file 206 also allows for aliasing and fragmentation of registers in order to improve efficiency and flexibility for register accessing.

[040] Instruction fetch stage 202 fetches the current instruction from a memory, e.g., an instruction buffer (not shown) or program memory 212, and forwards the instruction to instruction decode stage 204. Stage 204 decodes the instruction and sends inputs to register file 206 to access instruction data in appropriate registers. The accessed instruction data is sent to execution stage 208 in order for execution stage 208 to process the instruction. **FIG. 3** illustrates in block diagram form one example of inputs and output for register file 206. Thus, referring also to **FIG. 3**, for example, instruction decode stage 204 sends "processor mode inputs" and "source index inputs" to register file 206. Register file 206 uses

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

the inputs from instruction decode stage 204 to obtain an encoded address for accessing the desired register within register file 206 in order to execute the instruction. Thus, data from the register addressable by the encoded address is forwarded to execution stage 208.

[041] Moreover, the "processor mode inputs" and "write index inputs" can be sent to register file 206 by other components or control circuitry (not shown) to write data into a desired register within register file 206. For example, data received at "write data inputs" can be written into register 206 at an encoded address derived from the "processor mode inputs" and "write index inputs." This allows write back or retire logic 210 to send data for storage in register file 206 via the "write index inputs." The manner in which these inputs are used by register file 206 to obtain an encoded address is described in further detail below.

[042] Execution stage 208 can include any number of instruction execution units. Examples of execution units include arithmetic logic units (ALUs), load/store units, multiply and accumulate units (MACs), etc. Execution stage 208 also operates with program memory 212 to execute instructions. For example, for a load/store operation, execution stage 208 can store data into program memory 212 after processing an instruction using a load/store execution unit. In one example, instructions can be issued to execution stage 208 in-order and executed out-of order. For data processing system 200, results of the instructions are retired in-order to register file 206.

[043] Thus, in this example, execution stage 208 can forward data from executed instructions ("results data") to write back or retire logic 210 ("logic 210"). Logic 210 can forward results data back to register file 206 for storage. In particular, logic 210 writes back results of executed instructions from execution stage 208 to register file 206. Additionally, execution stage 208 can execute instructions out-of-order by using a re-order buffer. In one

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

example, retire logic 120 includes a re-order buffer and retires instruction data in the reorder buffer by sending the data to register file 206 for storage. The required data can then be used by subsequent instructions. As further explained below, register file 206 includes a plurality of write ports to receive write data (instruction data) from logic 210. Accordingly, logic 210 can retire one or more results (instruction data) for executed instructions in a same cycle such that the results are stored in register file 206.

[044] With further reference to **FIG. 3**, register file 206 is scalable to receive any number of inputs and output any number of outputs. Thus, in the example shown in **FIG. 3**, register file 206 includes four sets of input ports to receive processor mode inputs, source index inputs, write index inputs, and write data inputs. Each set of input ports receives a plurality of inputs (input 1 through input N). Register file 206 also includes a set of output ports to output source data outputs (output 1 through output N). In this example, register file 206 can receive inputs and provide outputs for a multiple issue data processing system. Specifically, register file 206 is capable of outputting and storing data for multiple instructions in a same cycle.

[045] For example, if two instructions require two ALU operations and each operation requires two source data inputs from register file 206, register file 206 can receive four processor mode inputs and four source index inputs in a same cycle to access and output data from four registers as source data outputs. This capability assumes there are no data dependencies, e.g., the second ALU operation does not require the result of the first ALU operation as an input. In particular, if the first ALU operation is A + B and the second ALU operation is C + D and the operand data for A through D is stored in four different registers of register file 206, N = 4 for the number of inputs and outputs for register file 206.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

Accordingly, register file 206 uses processor mode inputs 1 through 4 and source index inputs 1 through 4 to obtain encoded addresses for accessing the four registers holding the operand data A through D in register file 206 in order to output the operand data as source outputs 1 through 4 .

[046] Similarly, if two instructions can be retired from retire logic 210, register file 206 can store data from the two retired instructions in a same cycle through write data input 1 and write data input 2. For example, register file 206 uses processor mode input 1 and write index input 1 to obtain an encoded address for the register storage location within register file 206 for data received at write data input 1. In addition, register file 206 uses processor mode input 2 and write index input 2 to obtain an encoded address for the register storage location within register file 206 for data received at write data input 2. The manner of processing the above inputs and outputs is explained in further detail below.

[047] **FIG. 4** illustrates a detailed circuit diagram of one example of register file 206 of **FIG. 3**. In this example, register file 206 can operate in a multiple issue data processing system, more specifically, a dual issue data processing system. As shown, register file 206 includes a register file memory unit 400, a plurality of source address (read) encoders 402_1 through 402_4, and (write) address encoders 410_1 and 410_2. As described in further detail below, each of the encoders 402_1 through 402_4, 410_1, and 410_2 include mapping control logic for obtaining an encoded address using a mapping table, based on a processor mode and source index input, to access a register (or memory location) in register file memory unit 400.

[048] Register file memory unit 400 ("memory unit 400") is a single memory unit, examples of which include a static random access memory (SRAM) or a plurality of flip-

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

flops. Memory unit 400 is scalable and capable of having any arbitrary size containing, e.g., 16, 32, 48, or 64 registers, each capable of being addressed by an encoded address. Each register in memory unit 400 represents a general purpose register for access in particular processor mode and is addressable by an encoded address. In one example, an encoded address maps to an index referring to a general purpose register for access in a particular processor mode, as explained in further detail below.

[049] Additionally, each register is capable of having any arbitrary bit width size. For example, each register can be 32-bits wide or 64-bits wide. Because memory unit 400 is a single memory unit, the widths can be adjusted without affecting the encoded addressing scheme for memory unit 400. This minimizes extensive redesign if using different widths for memory unit 400. In other words, regardless of the bit width size, each register can be addressable by the same encoded address using the same encoded addressing scheme. This allows register file 206 to be expandable for 32-bit or 64-bit architectures with minimal redesign in which an efficient and simple encoded addressing scheme is used.

[050] Read encoders 402_1 through 402_4 receive processor_mode inputs and src1.index through src4.index inputs, respectively, and write encoders 410_1 and 410_2 receive processor_mode inputs and wr1.index and wr2.index inputs. Read encoders 402_1 through 402_4 map a source general purpose register index (via the index inputs) to an encoded address during a particular processor mode using a mapping control logic and mapping tables described below. The encoded address is used for accessing data in a specific register in memory unit 400 during the particular processor mode and to output the data as source data for executing instructions. In this example, memory unit 400 can output a plurality of source data (src1_data through src4_data) for at least two instructions.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[051] Read encoders 402_1 through 402_4 can either latch the encoded addresses in associated latches 404_1 through 404_4, respectively, or directly output encoded addresses to associated selectors 406_1 through 406_4, respectively, that also receive as inputs the output of latches 404_1 through 404_4. Latches 404_1 through 404_4 latch resultant encoded addresses for pipeline storage of the encoded address in the case that data for an instruction can be reused at the register or storage location of the latched encoded address. Selectors 406_1 through 406_4 select either the encoded address directly from the associated address encoders or from the associated latches in response to a signal (not shown) generated by the instruction decode stage 204. In one example, if one of the selectors 406_1 through 406_4 selects an encoded address from the associated read address encoder, the encoded address is also latched for pipeline storage. In this manner, the encoded address, as opposed to the source index, is latched for pipeline storage, thereby providing further processing efficiency.

[052] Likewise, write encoders 410_1 and 410_2 map a write index (via write index inputs) to an encoded address during a particular processor mode using a mapping control logic and mapping tables described below. The encoded address is used for accessing a specific register in memory unit 400 during the particular processor mode to store data associated with an executed instruction. In this example, a plurality of write data (wr0_data and wr1_data) can be written or stored into memory unit 400. For example, if two instructions can be retired or written back, wr0_data and wr1_data can be written or stored in memory unit 400 in a same cycle. Memory unit 400 can be scalable to have any number of write ports to which any number of data can be written or stored in memory unit 400 of register file 206.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[053] The mapping control logic and various examples of mapping tables for obtaining encoded addresses for accessing registers in memory unit 400 of register file 206 will now be described. Such mapping logic and mapping tables for obtaining encoded addresses are used by encoders 402_1 and 402_4, and 410_1 and 410_2 described above.

[054] **FIG. 5** illustrates a block diagram of one example of an address encoder 500 with a mapping control logic 502 and a mapping table 504 for obtaining encoded addresses to access registers or storage locations in memory unit 400 of **FIG. 4**. By using mapping control logic 502 and mapping table 504 to obtain encoded addresses for registers in memory unit 400, a more flexible, efficient, and convenient manner of accessing registers across different processor modes can be achieved. In the following examples, mapping control logic 502 and mapping table 504 can be implemented as circuitry, examples of which include programmable gate arrays (PGAs), field programmable gate arrays (FPGAs), or other like circuitry to obtain certain outputs based on varying inputs, as described below.

[055] Address encoder 500 can be representative of any one of encoders 402_1 through 402_4, and 410_1 and 410_2. In this example, latches and selectors are not shown for purposes of explanation. As shown in **FIG. 5**, mapping control logic 502 receives a processor mode input and a general purpose register (GPR) index input that correspond to the processor_mode inputs and (src.index and wr.index inputs), respectively, shown in **FIG. 4**. The GPR index input is capable of receiving 1 to N different indices that correspond to general purpose registers R_0 through R_{N-1}. For example, referring to **FIG. 6**, a block diagram is shown of one example of general purpose register (GPR) indices 600 ($index_1$ to $index_N$) corresponding to a plurality of general purpose registers (R_0 to R_{N-1}).

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[056] Mapping control logic 502 can thus receive an index, e.g., $index_1$, that indicates that general purpose register R_0 is to be accessed. Mapping control logic 502 can also receive a processor mode, e.g., mode 1, that indicates register R_0 for processor mode 1 is to be accessed in a corresponding register of memory unit 400. Mapping control logic 502 uses mapping table 504 that maps GPR indices for various processor modes to encoded addresses, which are used to access a corresponding register in memory unit 400 of register file 206. For example, referring to **FIG. 7**, a block diagram of one example of mapping table 504 is shown. In mapping table 504, any arbitrary number of GPR indices ($index_1$ to $index_N$) can be mapped to any arbitrary number of encoded addresses (encoded $address_1$ to encoded $address_M$) for any number of processor modes (mode 1 to mode K).

[057] After receiving the processor mode input and GPR index input, mapping control logic 502 obtains the appropriate encoded address using mapping table 504. In this process, inputs are provided to mapping table 504, which can include programmable hardware circuitry, to obtain a desired output (i.e., encoded address). The obtained encoded address is outputted to memory unit 400 in register file 206 in order to access a desired register. Depending on whether data is to be inputted or outputted, the register addressable by the obtained encoded address is accessed and data is either stored or outputted, accordingly. In this example, N is =, <, or > than M, wherein N and M are integers.

[058] More particularly, memory unit 400 can have less registers than the number of indices, which allows for efficient use of memory space. For instance, each encoded addressable memory location register in memory unit 400 is capable of mapping to multiple GPR indices or registers for access in different processor modes. Furthermore, memory unit 400 can have the same or more registers than the number of GPR indices to provide

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

fragmentation of registers in memory unit 400. For instance, registers in memory unit 400 addressable by an encoded address can be discontinuous and scattered throughout memory unit 400. Fragmentation is useful, e.g., when operating in emulation mode, such that registers can be isolated when not operating in normal modes. These features are explained in further detail below.

[059] **FIG. 8** illustrates a block diagram of one example of sixteen GPR indices 800 corresponding to sixteen general purpose registers (R_0 to R_{15}) for use by a data processing system or processor. These registers are associated with register indices ranging from 0000 to 1111 for the sixteen registers. In one example, the sixteen register indices (0000 to 1111) map to thirty-two encoded addresses (00000 to 11111) for addressing thirty-two registers in memory unit 400 across different modes.

[060] For example, referring to **FIG. 9**, a diagram is shown of one example of a mapping table 904 for the sixteen GPR indices 800 of **FIG. 8** mapping to thirty-two encoded addresses. Mapping table 904 contains thirty-two encoded addresses that map to sixteen GPR indices for processor modes 1 through N. In particular, each encoded address maps to one of the indices ranging from 0000 to 1111 based on at least one processor mode. Mapping table 904 thus allows a single memory unit, e.g., memory unit 400, to be used for accessing registers designated for different processor modes using the encoded addresses.

[061] This map can be used by encoders to obtain encoded addresses using register index inputs and processor mode inputs for accessing thirty-two registers in memory unit 400 of register file 206. This example, shown in **FIG. 9**, is based on data processing system 200 operating in N different processor modes. For example, mode 1 can be designated as a user mode and mode 2 can be designated as an interrupt mode. Modes 3

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

through 7 can be designated for other types of modes such as various exception handling modes.

[062] As illustrated in mapping table 904, registers R_0 through R_7 associated with indices ranging from 0000 to 0111 map to 5-bit encoded addresses ranging from 00000 to 00111. In this manner, accessing general purpose registers R_0 through R_7 is performed by accessing the registers in memory unit 400 addressable by encoded addresses ranging from 00000 to 00111. Moreover, accessing register R_1 associated with modes 1 - N is performed by accessing the register corresponding to the memory location in memory unit 400 addressable by encoded address 00001. For accessing general purpose registers R_8 through R_{12} in mode 1, the registers corresponding to the memory locations addressable by the encoded addresses ranging from 01000 to 01100 are used. For accessing general purpose registers R_8 and R_{12} in mode 2, the registers corresponding to the memory locations addressable by the encoded addresses ranging from 01101 to 10001 are used. A similar technique is used for accessing registers for modes 3 through N in memory unit 400 of register file 206, as shown in **FIG. 9**.

[063] Thus, unlike prior art register files, register file 206 uses a single memory unit to access registers across different processor modes. The registers and their corresponding memory locations are addressable by an encoded address corresponding to respective registers and processor modes. As a result, registers used for various processor modes can be accessed using a single memory unit without using "banked registers."

[064] Further examples of mapping tables are shown in **FIGS. 10A-10C** mapping 1 to N GPR indices to 1 to N-1 registers in 1 to M encoded address locations. In the following examples, N =, >, or < than M, wherein N and M are integers. **FIG. 10A**

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

illustrates one example of aliasing registers in memory unit 400 of register file 206. Aliasing allows multiple GPR indices to map to the same register in memory unit 400 of register file 206. As shown, $index_1$ corresponding to register R_0 maps to encoded address $location_1$ in memory unit 400 and $index_2$ and $index_3$ corresponding to registers R_1 and R_2 map to the same encoded address $location_2$ in modes 2 and 3, respectively.

[065] Aliasing allows for efficient use of memory space in memory unit 400 and provides a convenient manner of addressing registers in memory unit 400. For instance, data stored in encoded address $location_2$ can be shared between modes 2 and 3 without a data transfer taking place into encoded address $location_2$. Furthermore, because indices 2 and 3 for processor modes 2 and 3, respectively, map to the same register at encoded address $location_2$, a single register for two different processor modes can be used instead of requiring two separate registers. The mapping tables for aliasing described herein thus provide a simple manner of associating encoded addresses to GPR indices across different processor modes.

[066] **FIG. 10B** illustrates one example of aliasing and fragmenting registers in memory unit 400 of register file 206. Fragmentation allows for GPR indices to map to registers that are discontinuous in memory unit 400. As shown, $index_1$ corresponding to register R_0 maps to encoded address $location_1$ in memory unit 400 and $index_2$ and $index_3$ corresponding to registers R_1 and R_2 for mode 2 map to the same register at encoded address $location_2$. In this example, the top portion of memory unit 400 is not used for mapping encoded addresses to GPR indices. This unused portion of memory unit 400 can be used for other purposes within the data processing system or processor.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[067] Fragmentation also allows for flexible use of memory space within memory unit 400. In particular, during less often used modes, such as an emulation mode, it may be desirable for the data processing system or processor to use different registers not used by other modes. In this manner, data in the registers for the other modes are not destroyed. For instance, as shown in **FIG. 10C**, mode 1 may be an emulation mode, such that $index_1$ corresponding to register R_0 maps to the register in memory unit 400 at encoded address $location_1$. Encoded address $location_1$ is isolated or fragmented from the next register at encoded address $location_2$ and encoded address $location_3$.

[068] **FIG. 11** illustrates one example of a flow diagram for a method 1100 to output source data from register file 206 of **FIG. 4** using mapping table 904 as shown in **FIG. 9**. Initially, processor mode and source data index inputs are received (step 1102). For example, if an ALU instruction is to be executed in mode 2 that needs values A and B from general purpose registers R_8 and R_9, the processor mode input would be "mode 2" and the source data index inputs (src1.index and src2.index) would be 1000 and 1001 for registers R_8 and R_9.

[069] Next, an encoded address is obtained based on the received inputs by an address encoder (using mapping table 904) to output the obtained encoded address to a latch (step 1104). Using the example of memory map 904 above, a source index 1000 maps to encoded address 01000 and source index 1001 maps to encoded address 01001. A selector selects the encoded address stored in the latch or from the address encoder directly (step 1106). For instance, selectors 406_1 and 406_2 can select encoded addresses (01000 and 01001) from either latches 404_1 and 404_2, respectively, or address encoders 402_1 and 402_2 directly in which the encoded addresses are outputted to memory unit 400. The register addressable by

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

the outputted encoded address is accessed in memory unit 400 (step 1108). In particular, registers R_8 and R_9 can be addressed using the encoded addresses 01000 and 01001 to access and obtain data from registers R_8 and R_9. Finally, the data A and B at the addressed register are outputted (step 1110). That is, the data A and B stored in registers R_8 and R_9 at encoded addresses 01000 and 01001 are outputted as src1_data and src2_data from memory unit 400 to execution stage 204 to execute the ALU operation.

[070] The method shown in **FIG. 11** can also be implemented for multiple instructions in which four registers are accessed, assuming there are no data dependencies. For example, if two ALU instructions are required needing values A through D stored in registers R_8 through R_{11} for mode 2, the address encoders 402_1 through 402_4 obtain the encoded addresses 01000 through 01011 from the source indices 1000 through 1011 for registers R_8 through R_{11}. The data values A through D at storage locations addressable by the encoded addresses are outputted as src1_data through src4_data.

[071] **FIG. 12** illustrates one example of a flow diagram for a method 1200 to write data into register file 206 of **FIG. 4** using mapping table 904 as shown in **FIG. 9**. Initially, processor mode and write index inputs are received (step 1202). In method 1200, wr0_data and wr1_data received at write input ports are to be written into two registers (R_8 and R_9) of register file memory unit 400 during mode 2. Thus, the processor mode input would be "mode 2" and the wr0.index would be 1000 and the wr1.index would be 1001.

[072] Next, an encoded address is obtained based on the received inputs (step 1204). Using the example of memory map 904 above, wr0.index of 1000 maps to encoded address 01000 and wr1.index of 1001 maps to encoded address 01001. The register addressable by the encoded address is accessed in memory unit 400 (step 1206). In

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

particular, registers R_8 and R_9 can be addressed using the encoded addresses 01000 and 01001 for writing data to registers R_8 and R_9. Finally, the wr0_data and wr1_data are written into the accessed registers addressable by the encoded addresses 01000 and 01001.

[073] Thus, a register file has been described that can be a memory mapped register file. The register file described herein can be implemented for general computing devices, examples of which include microprocessors, processors, central processing units (CPUs), application specific integrated circuits (ASICs), system on a chips (SOCs), embedded processors or systems, micro-controllers, and other computing devices. Moreover, the register file can be implemented for multi-stage and variable stage pipelining architectures that operate in different processor modes.

[074] Furthermore, in the foregoing specification, the invention has been described with reference to specific exemplary embodiments and implementations thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com